PCT/JP2004/009000

02.7.2004

日本国特許庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

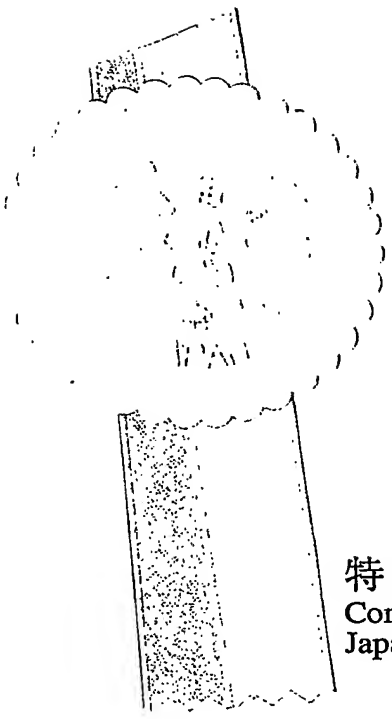
This is to certify that the annexed is a true copy of the following application as filed with this Office.

出願年月日
Date of Application: ２００３年　６月２７日

出願番号
Application Number: 特願２００３－１８５４８１

[ST. 10/C]: [ＪＰ２００３－１８５４８１]

出願人
Applicant(s): アイピーフレックス株式会社



２００４年　８月１２日

特許庁長官
Commissioner,
Japan Patent Office

小川洋

出証番号　出証特２００４－３０７１９６０

【書類名】　特許願
【整理番号】　030136P502
【あて先】　特許庁長官殿
【国際特許分類】　G06F　9/45
【発明者】
　【住所又は居所】　東京都品川区上大崎二丁目２７番１号　アイピーフレックス株式会社内
　【氏名】　志村　大
【特許出願人】
　【識別番号】　500238789
　【氏名又は名称】　アイピーフレックス株式会社
【代理人】
　【識別番号】　100102934
　【弁理士】
　【氏名又は名称】　今井　彰
【手数料の表示】
　【予納台帳番号】　050728
　【納付金額】　21,000円
【提出物件の目録】
　【物件名】　明細書　1
　【物件名】　図面　1
　【物件名】　要約書　1
【プルーフの要否】　要

【書類名】　明細書

【発明の名称】　並列処理システムの生成方法

【特許請求の範囲】

【請求項１】　並列に動作する複数の要素を備えたシステムにより、同期して独立に行われる複数の並列処理をそれぞれ規定した複数の並列記述を有する定義ファイルであって、前記複数の並列記述は、少なくとも１つの他の並列処理の出力データを少なくとも１つの入力データとする並列処理を示す並列記述を含み、複数の入力データを備えた前記並列記述においては、それら複数の入力データは、当該システムに入力されてからのレイテンシーが同一である定義ファイルが記録されているコンピュータ読み取り可能な記録媒体。

【請求項２】　請求項１において、前記複数の並列記述は、前記複数の要素が動作するクロックに同期して実行される前記複数の並列処理をそれぞれ規定する、コンピュータ読み取り可能な記録媒体。

【請求項３】　独立に行われる複数の並列処理をそれぞれ規定した複数の並列記述を有する定義ファイルに基づき、並列に動作する複数種類の要素を備えた並列処理システムを生成する方法であって、

前記定義ファイルの複数の並列記述には、少なくとも１つの他の並列処理の出力データを少なくとも１つの入力データとする並列処理を示す並列記述が含まれており、

前記複数種類の要素の情報を記録したハードウェアライブラリに基づき、前記並列処理を実行するための回路構成であって、少なくとも１つの前記要素を含む回路構成を生成する第１の工程と、

前記回路構成に複数のデータが入力される場合は、それら複数の入力データの、当該並列処理システムに入力されてからのレイテンシーが一致するように遅延要素を加える第２の工程とを有する並列処理システムの生成方法。

【請求項４】　請求項３において、前記並列処理システムは、前記複数種類の要素の接続を変えることにより異なるハードウェア構成を実現可能であり、

前記第２の工程により生成された前記回路構成を含む前記ハードウェア構成を前記複数種類の要素の接続情報を含むハードウェア構成情報として出力する工程

をさらに有する並列処理システムの生成方法。

【請求項５】　請求項３において、前記複数種類の要素は、単体で前記並列処理を処理可能な程度の規模の複数種類の演算ユニットである、並列処理システムの生成方法。

【請求項６】　請求項３において、前記複数種類の要素は、バイトあるいはワード単位で異なった演算を実行可能な複数種類の演算ユニットである、並列処理システムの生成方法。

【請求項７】　請求項３において、前記第２の工程では、前記複数の要素が動作するクロックに同期した前記遅延要素を加える、並列処理システムの生成方法。

【請求項８】　請求項３において、前記並列記述により規定された並列処理が、他の並列記述により規定された並列処理の少なくとも一部と同じ共通処理を含む場合は、前記第１の工程では、前記共通処理に同一の前記要素を含む共通構成を割り当て、前記第２の工程では、前記共通構成に含まれない前記要素に対して遅延要素を加える、並列処理システムの生成方法。

【請求項９】　独立に行われる複数の並列処理をそれぞれ規定した複数の並列記述を有する定義ファイルに基づき、並列に動作する複数種類の要素を備えた並列処理システムを生成する生成装置であって、

前記定義ファイルの複数の並列記述には、少なくとも１つの他の並列処理の出力データを少なくとも１つの入力データとする並列処理を示す並列記述が含まれており、

前記複数種類の要素の情報を記録したハードウェアライブラリに基づき、前記並列処理を実行するための回路構成であって、少なくとも１つの前記要素を含む回路構成を生成する第１の手段と、

前記回路構成に複数のデータが入力される場合は、それら複数の入力データの、当該並列処理システムに入力されてからのレイテンシーが一致するように遅延要素を加える第２の手段とを有する並列処理システムの生成装置。

【請求項１０】　請求項９において、前記並列処理システムは、前記複数種類の要素の接続を変えることにより異なるハードウェア構成を実現可能であり、

前記第２の手段により生成された前記回路構成を含む前記ハードウェア構成を前記複数種類の要素の接続情報を含むハードウェア構成情報として出力する手段をさらに有する並列処理システムの生成装置。

【請求項１１】　独立に行われる複数の並列処理をそれぞれ規定した複数の並列記述を有する定義ファイルに基づき、並列に動作する複数種類の要素を備えたシステムを設計する設計プロセスをコンピュータにより実行可能とするプログラムであって、

前記定義ファイルの複数の並列記述には、少なくとも１つの他の並列処理の出力データを少なくとも１つの入力データとする並列処理を示す並列記述が含まれており、

前記複数種類の要素の情報を記録したハードウェアライブラリに基づき、前記並列処理を実行するための回路構成であって、少なくとも１つの前記要素を含む回路構成を生成する第１の工程と、

前記回路構成に複数のデータが入力される場合は、それら複数の入力データの、当該並列処理システムに入力されてからのレイテンシーが一致するように遅延要素を加える第２の工程とを有するプログラム。

【請求項１２】　独立に行われる複数の並列処理をそれぞれ規定した複数の並列記述を有する定義ファイルに基づき、並列に動作する複数種類の要素を備えたシステムをシミュレートする方法であって、

前記定義ファイルの複数の並列記述には、少なくとも１つの他の並列処理の出力データを少なくとも１つの入力データとする並列処理を示す並列記述が含まれており、

前記定義ファイルに含まれる複数の並行処理を同期して実行する工程を有し、この実行する工程では、複数の入力データを備えた前記並列記述を実行する際には、それら複数の入力データに、当該システムに入力されてからのレイテンシーが同一のデータを用いる、シミュレーション方法。

【請求項１３】　独立に行われる複数の並列処理をそれぞれ規定した複数の並列記述を有する定義ファイルに基づき、並列に動作する複数種類の要素を備えたシステムをシミュレートするシミュレータであって、

前記定義ファイルの複数の並列記述には、少なくとも１つの他の並列処理の出力データを少なくとも１つの入力データとする並列処理を示す並列記述が含まれており、

前記定義ファイルに含まれる複数の並行処理を同期して実行する手段を有し、この実行する手段では、複数の入力データを備えた前記並列記述を実行する際に、それら複数の入力データに、当該システムに入力されてからのレイテンシーが同一のデータを用いる、シミュレータ。

【請求項１４】　独立に行われる複数の並列処理をそれぞれ規定した複数の並列記述を有する定義ファイルに基づき、並列に動作する複数種類の要素を備えたシステムをコンピュータによりシミュレートするプログラムであって、

前記定義ファイルの複数の並列記述には、少なくとも１つの他の並列処理の出力データを少なくとも１つの入力データとする並列処理を示す並列記述が含まれており、

前記定義ファイルに含まれる複数の並行処理を同期して実行する工程を有し、この実行する工程では、複数の入力データを備えた前記並列記述を実行する際に、それら複数の入力データに、当該システムに入力されてからのレイテンシーが同一のデータを用いる、プログラム。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は、並列処理システムの設計に関するものである。

【０００２】

【従来の技術】

ＬＳＩあるいはＡＳＩＣを設計するために幾つかの言語が用いられている。Ｃ言語などの抽象度の高い高級言語と称されるものは、手続き的レベルの言語であり、１命令毎にいかに処理全体が順番に実行されていくかを示すのに適している。このレベルの記述は、一般的にハードウェア依存性がなく、適当なコンピュータで受け入れられるアプリケーションプログラムであり、ＬＳＩの仕様、またはＬＳＩにおいて実行する処理全体を一般的に記述するために用いられる。Ｖｅｒ

ｉｌｏｇ－ＨＤＬあるいはＶＨＤＬなどのハードウェア記述言語（ＨＤＬ）は、ＲＴＬと称されることもありレジスタートランジスタレベルで、特定のハードウェアにより特定の命令が実行されているデータパスやパスを駆動するシーケンスを記述するために用いられる。

【０００３】

【特許文献１】

特開平１０－１１６３０２号公報

【０００４】

【発明が解決しようとする課題】

アルゴリズムは、問題を解くために明確に規定され、順序付けられた有限個の規則からなる集合として定義されており、従来、並列処理は、アルゴリズムにより記述された処理全体（アプリケーション）を順番に進める上で、独立して実行可能な部分（処理）を並列に実行し、処理時間を短縮する目的で用いられる。予め並列処理に適したハードウェアリソースを備えたシステムでアプリケーションを実行する場合は、コンパイラなどにより並列処理ができる部分は並列化され、実行速度を向上することがトライされる。

【０００５】

また、特定のアプリケーションの実行を目的としたハードウェアを設計する場合は、独立して実行できる部分を並列に処理するように回路を設計して処理時間の短縮を目指す。特開平１０－１１６３０２号公報に記載された技術は、並列処理および同期通信等により実行時間が定数でない処理を記述可能なＨＤＬにより回路を設計する方法である。同期通信とは、２つのファンクションを並行に実行する際に、それらに含まれるプロセスを送信側が準備できるまで受信側が待ち、通信が完了した後にプロセスが進行する。したがって、これらのファンクションは並列に記述されても独立して実行されず、実行時間が可変の処理となる。一方、同期通信を行わない処理は並列処理として独立して行われる。これらは、ソース言語で与えられた処理全体の内、ソース上では並列に実行するように記載された処理を、ハードウェア設計において並列に、あるいは同期通信を用いて実行して実行サイクル数を短縮することを目的とした技術である。

【０００６】

近年、ＬＳＩを構成する回路の一部をソフトウェアにより再構成できるハードウェアが提供されている。さらに、国際公開ＷＯ０３／００７１５５号には、再構成する基本単位を、ゲートレベルからＡＬＵなどのある程度の規模の演算機能を備えた演算ユニットにして、複数種類の演算ユニットをマトリクス状に配置し、再構成に要する時間を短縮することが開示されている。複数の演算ユニットがマトリクス状に配置されたシステムは、それぞれの演算ユニットが並列に処理を実行できるので、膨大な数の並列処理に適したハードウェアリソースを備えたシステムと捉えることが可能である。しかしながら、この種の並列処理に適したシステムを設計するのに適した設計システムは提供されていない。

【０００７】

Ｃ言語などのソフトウェア設計に適した高級言語は、アルゴリズムを、それに含まれる規則を時間的な順番に処理することを前提としている。したがって、プログラムカウンタを進めて命令がシーケンシャルに実行されるように構成されており、シーケンシャルではない並列という概念を導入することは難しい。命令を並列的に記述することが許容されたとしても、それは、時間的な順番に齟齬をきたさない範囲で、独立して実行できる処理を空間的に並列に広げて実行できる程度であり、並列処理に適したハードウェアリソースを積極的に使用することはできない。さらに、高級言語であれば、ハードウェアに依存しない命令が記述されるために、並列に記述した命令がハードウェアで実際に開始されたり、終了したりするタイミングは不明である。したがって、処理できる範囲を空間的に広げても、設計者はハードウェア上で実際にどのように並列処理が実行されるのかは定義できないし、把握することもできない。

【０００８】

ＨＤＬは独立して動作する回路構成を記述するので、本来、並列処理を記述するものである。また、ハードウェアが明確になるので、処理が実行されるタイミングを調べたり、調整できる。このため、高級言語で与えられたアルゴリズムを実現するＨＤＬを記述することができる。しかしながら、逆に、特定のハードウェアを前提として記述されるので、汎用性はなく、ハードウェアが異なれば同一

のアルゴリズムを実現することは不可能である。また、ＨＤＬが対象としているハードウェアが分からなければＨＤＬに含まれているアルゴリズムを理解することもできない。

【０００９】

高級言語を特定のハードウェア用にコンパイルした結果として得られるマイクロプログラムのレベルでは、完全に独立して実行可能な命令を並列に記述するＶＬＩＷ技術、複数の命令を同時にフェッチして並列実行できる命令を見つけ出して実行するスーパスケーラ技術がある。これらは時間的に並んだ処理のうち、それを実行するために用意された複数のパイプラインで空間的に並列に実行できるものを並列に処理して実行速度を改善する技術であり、時間的な順番に齟齬がおきない範囲を空間的に広げる点では高級言語と変わりはない。すなわち、マイクロプログラムもプログラム言語であり、プログラムカウンタを進めて命令を順番に実行するシーケンシャルな処理はＶＬＩＷやスーパスケーラにおいても同様に必要とされる。さらに、マイクロプログラムは、特定のハードウェアを前提としており、ＨＤＬと同様に汎用性は少ない。

【００１０】

このように、プログラムカウンタを進めて実行するプログラム言語では、高級言語であれば汎用性があり、ソフトウェア設計は行い易いが、並列処理に展開できる部分はプログラムカウンタを進めても独立して実行できる処理に限られてしまい、多数の並列処理に適した演算ユニットを有効に利用することは難しい。ハードウェア依存のない高級言語では、さらに、並列処理のタイミングも分からないので、マトリクス状に配置された多数の演算ユニットを並列に動作させてアプリケーションを効果的に実行する設計は不可能である。一方、ＨＤＬは並列処理の記述は可能であるが、アルゴリズムを記述したとしても特定のハードウェアを前提としてアルゴリズムが記述できるだけなので、特定のハードウェアの知識が必要である。したがって、マトリクス状に配置された多種類の演算ユニットの機能および入出力のタイミングなどをソフトウェア技術者が理解し、アプリケーションをＨＤＬで設計することは無理がある。

【００１１】

【課題を解決するための手段】

そこで、本発明においては、並列に動作する複数の要素を備えたシステムにより、同期して独立に行われる複数の並列処理をそれぞれ規定した複数の並列記述を有する定義ファイルであって、複数の並列記述は、少なくとも１つの他の並列処理の出力データを少なくとも１つの入力データとする並列処理を示す並列記述を含み、複数の入力データを備えた並列記述においては、それら複数の入力データは、当該システムに入力されてからのレイテンシーが同一である定義ファイルを提供する。この定義ファイルにより、ハードウェア依存性のない記述により、プログラムカウンタの必要のない形態でアルゴリズムを定義することができる。すなわち、この定義ファイルにより、従来の高級言語と類似した記述ではあるが、時間順序性のない並列記述により、アルゴリズムに含まれる順序を時間的にではなく空間的に記述できる。

【００１２】

この定義ファイルは、処理させる仕事（アプリケーション）の手順を定義している点では、従来のプログラム言語と共通する。しかしながら、プログラム言語は、記述されたコンピュータに対する命令が、原則として最初から順番に、すなわち、時間的経過と共に実行されるものとして記述している。したがって、プログラム言語により定義されたアルゴリズムを実行するためには、実行する命令の順番を示すプログラムカウンタによる制御が必要となる。一方、記述された命令がプログラムカウンタにより順番に実行されるので、変数を含む命令を実行する際は、その命令に時間的に先行して実行される命令により変数の状態は一義的に決まり、定義されたアルゴリズムが不安定になることはない。

【００１３】

一方、本発明の定義ファイルは、同期して独立に行われる複数の並列処理をそれぞれ規定した複数の並列記述を有する。したがって、本発明の定義ファイルに定義された並列処理を実行するには、時間的な順番を示すプログラムカウンタは必要としない。そして、他の並列処理の出力データを入力データとする並列処理を示す並列記述を含むことによりアルゴリズムを定義できる。しかしながら、変数を含む並列記述では、変数の経過が一義的に定まらないとアルゴリズムが不安

定になる可能性がある。そこで、複数の入力データを備えた並列記述においては、それら複数の入力データは、当該システムに入力されてからのレイテンシーが同一であると解することにより、変数が不安定になることを防止している。したがって、本発明の定義ファイルでは、並列記述毎に変数が一義的に定まり、アルゴリズムを並列記述により正確に定義できる。また、本発明の定義ファイルでは、並列記述にしたがって並列処理は完全に独立して実行され、同期通信などの余分な処理がなくても、また、ハードウェアを特定することによりタイミング問題を解決しなくても、アルゴリズムを正確に記述できる。

【００１４】

本明細書において、入力データのレイテンシーが同一とは、それらの入力データが、同期してシステムにロードされたデータ群またはその同期してロードされたデータ群のいずれかが少なくとも１つの並列処理により処理されたデータ群に含まれることを言う。すなわち、レイテンシーが同一の入力データは、システムに入力されたタイミングが同一のデータ、またはそのデータが加工されたものである。

【００１５】

本発明の定義ファイルはＨＤＬと同様に複数の並列記述からなり、複数の並列記述により並列処理を実行するためにプログラムカウンタが不要である。したがって、定義ファイルは、ハードウェアを記述していると言える。さらに、定義ファイルには、他の並列処理の出力データを入力データとする並列処理を示す並列記述を含み、複数の入力データがある場合もそれらの状態は一義的に決まる。したがって、本発明により、アルゴリズムを正確に、そして見て分かるように定義できるハードウェア記述言語を提供できる。さらに、ハードウェアの詳細が分からなくても入力データは一義的に決まるので、実際に並列処理を行うハードウェアの詳細な情報あるいは知識は不要である。したがって、本発明の定義ファイルは、実際のハードウェアに依存せず、特定のハードウェアを前提にしないでハードウェアを記述できる。このため、ハードウェアインディペンデントで、極めて汎用的なハードウェア記述言語である。したがって、ソフトウェア技術者が簡単にＬＳＩ、特に、並列処理要素を多数含んだＬＳＩの設計あるいは生成するのに

適したツールあるいは記述となる。

【００１６】

さらに、本発明の定義ファイルに含まれる複数の並列記述を、それらの入力データのレイテンシーの順番に並べると、複数の並列記述が、それに含まれる変数の時間経過の順番で並ぶことになる。この状態は、命令が実行される順番に並んだプログラムと同じである。したがって、ソフトウェア技術者は、普通にプログラムを作成するのと同じ感覚で、本発明の定義ファイルを作成することが可能である。この点でも、定義ファイルは、ソフトウェア技術者が簡単にＬＳＩ設計を行うのに正に適した設計あるいは生成するのに適したツールとなる。そして、本発明の定義ファイルにより、並列処理が可能な複数の要素により、アルゴリズムを空間的に割り付けることが可能となり、多数の並列に動作する要素を備えたシステムを有効に活用してアプリケーションを高速で実行できる。

【００１７】

本発明の定義ファイルにおいては、並列記述毎に、入力データのレイテンシーが同一と判断され、タイミングが調整される。したがって、並列記述により規定される並列処理の単位で入力データのタイミングが調整されることになる。１つの並列処理が複数の演算素子、例えば、トランジスタなどで構成される回路で実行される場合は、タイミングの調整は並列処理単位より小さな単位、例えば、トランジスタ間で行うことが望ましいかもしれない。しかしながら、ＡＬＵなどのある程度の規模の演算機能を備えた演算ユニットがマトリクス状に配置されたシステムにおいては、並列記述に対して１つまたは数個の演算ユニットを割り当てることにより並列処理を実行できる。したがって、並列記述単位でタイミングを調整するためには、遅延用の演算ユニットを加えれば良く、本発明の定義ファイルは、複数種類の演算ユニットを接続してアプリケーションに適したハードウェア構成を実現する並列処理システムを記述するのに適している。このため、本発明の定義ファイルにより、並列処理システムを効率的に設計および開発することができ、また、シミュレートすることが可能となる。

【００１８】

本発明の定義ファイルは、コンピュータ読み取り可能な記録媒体に記録して提

供することができ、定義ファイルに基づき、コンピュータを用いて、並列に動作する複数種類の要素を備えた並列処理システムを生成することができる。定義ファイルの複数の並列記述には、少なくとも１つの他の並列処理の出力データを少なくとも１つの入力データとする並列処理を示す並列記述が含まれている。したがって、複数種類の要素の情報を記録したハードウェアライブラリに基づき、並列処理を実行するための回路構成（ハードウェア構成）であって、少なくとも１つの要素を含む回路構成を生成する第１の工程と、並列記述が複数の入力データを備えており、並列処理を実行する回路構成に複数のデータが入力される場合は、それら複数の入力データの、並列処理システムに入力されてからのレイテンシーが一致するように遅延要素を加える第２の工程とにより、定義ファイルに定義されたアルゴリズムを複数種類の要素により空間的に割り付けでき、アプリケーションを実行する並列処理システムを生成できる。

【００１９】

特定のハードウェアに関する情報が格納されたハードウェアライブラリにより各種類の要素で消費される時間が分かる。したがって、並列記述が複数の入力データを備えている場合に、それらの複数の入力データのレイテンシーが一致するように、特定のハードウェアを前提とした遅延要素を加えることができ、特定のハードウェアに定義ファイルに定義されたアルゴリズムを正確に割付できる。複数の要素がクロックに同期して処理を実行するハードウェアであれば、ハードウェアライブラリには、複数種類の要素毎に消費されるサイクル数を含む情報が格納されており、第２の工程では、複数の要素が動作するクロックに同期した遅延要素を加えられる。

【００２０】

並列処理システムは、ハードウェア構成が固定されるものであってもよい。並列処理システムは、複数種類の要素の接続を変えることにより異なるハードウェア構成を実現可能なものであってもよい。第２の工程または手段により生成されたハードウェア構成を複数種類の要素の接続情報を含むハードウェア構成情報として出力する工程を設けることにより、再構成用の情報が得られる。

【００２１】

本発明の並列処理システムの生成方法は、並列に動作する複数種類の要素を備えたシステムを生成する並列処理システム生成装置、例えばコンパイラとしても提供することが可能である。このコンパイラは、定義ファイルに基づき、並列に動作する複数種類の要素を備えた並列処理システムを生成する生成装置であって、複数種類の要素の情報を記録したハードウェアライブラリに基づき、並列処理を実行するための回路構成であって、少なくとも１つの要素を含む回路構成を生成する第１の手段と、回路構成に複数のデータが入力される場合は、それら複数の入力データの、当該並列処理システムに入力されてからのレイテンシーが一致するように遅延要素を加える第２の手段とを有する。また、コンパイラとしての機能を、並列処理システム生成プログラムあるいはプログラム製品として適当な記録媒体に記録したり、コンピュータネットワークを用いて提供することが可能である。この並列処理システム生成用のコンパイラプログラムは、上述した第１の工程と、第２の工程とを実行可能な命令を有するプログラムである。

【００２２】

上述したように、本発明の定義ファイルは、複数種類の要素が、単体で並列処理を処理可能な程度の規模の複数種類の演算ユニットである並列処理システムを生成するのに適している。したがって、ハードウェアライブラリには、単体で並列処理を処理可能な程度の規模の複数種類の演算ユニットの情報を用意することが望ましい。また、複数種類の要素はビット単位で演算するものであっても良い。しかしながら、定義ファイルに記述される並列処理は一般にバイトあるいはワード単位のデータ処理である。したがって、ハードウェアライブラリには、バイトあるいはワード単位で異なった演算を実行可能な複数種類の演算ユニットの情報を用意し、複数の演算ユニットが配置された並列処理システムを生成することが望ましい。

【００２３】

予め複数種類の演算ユニットがマトリクス状に配置され、演算ユニットを接続するネットワークあるいは回路配線の構成を変更することにより、異なるハードウェア構成にすることができる再構成可能な並列処理システムに対しては、コンパイラまたはコンパイラプログラムにより、定義ファイルの内容を実行するのに

適した実際のハードウェア構成情報を出力することができる。

【００２４】

定義ファイルに基づいて並列処理システムを生成する際には、幾つかの最適化を施すことができる。並列記述により規定された並列処理が、他の並列記述により規定された並列処理の少なくとも一部と同じ共通処理を含む場合は、第１の工程では、共通処理に同一の要素を含む共通構成を割り当て、第２の工程では、共通構成に含まれない要素に対して遅延要素を加える。これにより、要素や、要素を繋ぐ配線などのハードウェアリソースの消費を抑制できる。

【００２５】

並列処理システムを構成する複数種類の演算ユニットは、外部入力により処理を変動する手段を備えていても良い。それに対応して、複数の並列記述には、外部入力により処理が変動する並列処理を記述する並列記述を含められるようにすることが望ましい。ネットワークあるいは回路配線を変更しないでも処理内容を変更することができる並列処理システムの設計が可能となる。ネットワークあるいは回路配線と共に演算ユニットの処理内容を変更することも可能であり、さらにフレキシブルにリコンフィグラブルな並列処理システムの設計が可能となる。

【００２６】

本発明の定義ファイルに基づき、並列に動作する複数種類の要素を備えたシステムをシミュレートすることが可能である。定義ファイルに基づき、並列に動作する複数種類の要素を備えたシステムをシミュレートする方法およびシミュレータは、定義ファイルに含まれる複数の並行処理を同期して実行する工程または手段を有し、この実行する工程では、複数の入力データを備えた並列記述を実行する際に、それら複数の入力データに、当該システムに入力されてからのレイテンシーが同一のデータを用いる。また、定義ファイルに基づき、並列に動作する複数種類の要素を備えたシステムをコンピュータによりシミュレートするためのプログラムあるいはプログラム製品も本発明により提供され、ＣＤ－ＲＯＭなどの適当な記録媒体に記録したり、コンピュータネットワークを介して提供することができる。

【００２７】

【発明の実施の形態】

図１に、本発明の定義ファイルを用いてハードウェアを設計する過程を示してある。定義ファイル１はＤＩＤＬ（Device Independent Description Language）と称されており、コンパイラ２によりハードウェアライブラリ３の情報を参照して、ライブラリ３に格納されたハードウェアを用いたハードウェア構成４に変換される。ハードウェア構成４は、ＤＤＤＬ（Device Dependent Description Language）と称されている。コンパイラ２は汎用的なコンピュータ９を用いて実現されており、ＤＩＤＬ１を解釈してＤＤＤＬ４を出力する並列システム設計用のプログラム５がインストールされることによりコンパイラとして機能する。したがって、ＤＩＤＬ１は、コンピュータ読み取り可能な記録媒体６、例えばＣＤ－ＲＯＭあるいは磁気ディスクなどに記録されて提供される。インターネットなどのコンピュータネットワークによる通信を用いて提供することも可能であり、提供されたＤＩＤＬ１は、コンピュータ９の一部となる適当な記録媒体に記録されて使用される。

【００２８】

図２に、ハードウェアライブラリ３に要素の情報が格納されている並列処理システムの一例を示してある。この並列処理システムは、本出願人の国際出願公開公報ＷＯ０３／００７１５５号に開示されている再構成可能なプロセッサ（ＲＰ、Reconfigurable Processor）である。このＲＰ２０は、プログラムなどによって与えられる命令セットに基づきエラー処理を含めた汎用的な処理を行う汎用な構成の基本プロセッサ２１と、マトリクス状に配置された演算あるいは論理エレメントにより特定のデータ処理に適合したデータフローあるいは擬似データフローがバリアブルに形成されるＡＡＰ（Adoptive Application Processor）部あるいはＡＡＰユニット（以降ではＡＡＰ）５０と、このＡＡＰ５０からの割り込み処理を制御する割り込み制御部２２と、ＡＡＰ５０に作動用のクロック信号を供給するクロック発生部２８と、このＲＰ２０で提供可能な演算回路のフレキシビリティーをさらに向上するためのＦＰＧＡ部２７と、外部に対するデータの入出力を制御するバス制御部２９とを備えている。基本プロセッサ２１とＡＡＰ５０は、これらの間でデータを交換可能なデータバス２４ａと、基本プロセッサ２１

からＡＡＰ５０の構成および動作を制御するための命令バス２４ｂとにより接続されている。また、ＡＡＰ５０から割り込み制御部２２に信号線２５を介して割り込み信号が供給され、ＡＡＰ５０における処理が終了したり、処理中にエラーが発生したときはＡＡＰ５０の状態を基本プロセッサ２１にフィードバックできるようになっている。

【００２９】

ＡＡＰ５０とＦＰＧＡ２７との間もデータバス２６により接続されており、ＡＡＰ５０からＦＰＧＡ２７にデータを供給して処理を行い、その結果をＡＡＰ５０に返せるようになっている。さらに、ＡＡＰ５０は、ロードバス２３ａおよびストアバス２３ｂによってバス制御ユニット２９と接続されており、ＲＰ２０の外部のデータバスとの間でデータを交換できるようになっている。

【００３０】

図３にＡＡＰユニット５０の概要を示してある。ＡＡＰユニット５０は、複数の算術および／または論理演算を行う論理ブロック、論理ユニットあるいは論理要素（以降ではエレメント）がマトリクス状に配置されたマトリクス部５１と、そのマトリクス部５１に対してデータを供給する入力バッファ５２と、マトリクス部５１から出力されるデータを格納する出力バッファ５３を備えている。これら入力バッファ５２および出力バッファ５３は、それぞれ４つの小容量の入力メモリにより構成されており、アクセス調停ユニット５４を介して入出力バス２３ａおよび２３ｂに接続される。

【００３１】

マトリクス部５１が、データパスあるいはデータフローを再構成可能な並列処理システムの中心となる集積回路区画であり、複数種類の演算ユニットであるエレメント５５が縦方向に４つのラインを構成するようにアレイ状あるいはマトリクス状に配置されている。このマトリクス部５１は、これらのエレメント５５の間に配置された、横方向に延びた行配線群５７と、縦方向に延びた列配線群５８とを備えている。列配線群５８は、列方向に並んだ演算ユニット５５の左右に分かれて配置された配線群５８ｘおよび５８ｙが１対になっている。行配線群５７および列配線群５８との交点にはスイッチングユニット５９が配置されており、

行配線群５７の任意のチャンネルを、列配線群５８の任意のチャンネルに切り替えて接続できるようになっている。各々のスイッチングユニット５９は、設定を記憶するコンフィグレイションＲＡＭを備えており、プロセッサ部２１から供給されるデータによりコンフィグレイションＲＡＭの内容を書き換えることにより、行配線群５７と列配線群５８の接続を動的に任意に制御できる。このため、このマトリクス部５１においては、複数のエレメント５５の全部あるいは一部が配線群５７および５８により接続されて形成されるデータフローの構成を任意に動的に変更することができる。

【００３２】

ＲＰ２０においては、エレメント５５が並列に動作する要素であり、各種類のエレメント５５の機能、遅延、入出力データの条件などの情報がハードウェアライブラリ３に格納されている。エレメント５５は、クロック発生部２８から供給されるクロック信号に同期して稼動するので、ハードウェアライブラリ３には各種類のエレメント毎に、入力データを処理して出力するために消費されるサイクル数が遅延情報として格納されている。さらに、各種類のエレメント５５の配置と、配線群５７および５８、スイッチングユニット５９の情報もハードウェアライブラリ３に格納されており、コンパイラ２からは、ＤＩＤＬ１に定義されたアルゴリズムを実現するための、エレメント５５の接続情報（データフロー構成）がハードウェア構成情報（ＤＤＤＬ）４として出力される。このため、ＤＤＤＬ４に従ってエレメント５５が配線群５７および５８で接続されるようにマトリクス部５１を制御することにより、ＤＩＤＬ１に定義されたアルゴリズムをマトリクス部５１にエレメント５５により空間的に割り付けることが可能となる。

【００３３】

各エレメント５５は、１組の列配線群５８ｘおよび５８ｙのそれぞれから入力データを選択するための１組のセレクタ５４と、選択された入力データに特定の算術および／または論理演算処理を施し、出力データとして行配線群５７に出力する内部データパス部５６を備えている。そして、本例のマトリクス部５１には、各行毎に異なる処理を行うための内部データパス部５６を備えた種類の異なるエレメント５５が並んで配置されている。例えば、第１行目に配列されたエレメ

ント５５は、入力バッファ５２からのデータを受信する処理に適したデータパス部（ＬＤ）５６ｉを備えている。第２行目に配置されたエレメント５５ａは、入力バッファ５２に外部デバイスからデータを書き込むためのエレメントであり、ブロックロードするためのアドレスを発生するのに適した内部データパスを具備するデータパス部（ＢＬＡ）５６ａを備えている。マトリクス５１を構成する全てのエレメント５５は、内部データパスの構成あるいは初期値などがある程度変更できるようになっており、その設定は各々のエレメント５５のコンフィグレイションＲＡＭに基本プロセッサ２１から制御信号２４ｂにより指示される。

【００３４】

第３行目に配置されたエレメント５５ｂは、入力ＲＡＭの各々より所望のデータをマトリクス部５１へロードする入力読み出しアドレスを発生するデータパス部（ＬＤＡ）５６ｂを備えている。第４行目および第５行目に配列されたエレメント５５ｃは、算術演算および論理演算に適したデータパス部（ＳＭＡ）５６ｃを備えている。このデータパス部５６ｃは、たとえば、シフト回路、マスク回路、論理演算ユニットＡＬＵおよびＡＬＵで処理する演算をセットするコンフィグレイションＲＡＭを備えている。したがって、プロセッサ２１が書き込んだ命令により、マトリクス部５１へ入力されたデータを加算あるいは減算したり、比較したり、論理和あるいは論理積を取ったりすることができ、その結果がエレメント５５の出力信号として出力される。

【００３５】

その下の行に配列されたエレメント５５ｄは、データが伝送されるタイミングを遅延する処理に適したデータパス部（ＤＥＬ）５６ｄを備えている。その下の行に配列されたエレメント５５ｅは、乗算器などを含む乗算処理に適したデータパス部（ＭＵＬ）５６ｅを備えている。さらに異なるエレメント５５ｆとしては、マトリクス部５１の外部に用意されたＦＰＧＡ２７とのインターフェイス用のデータパス部５６ｆを備えたエレメントも用意されており、データをいったんＦＰＧＡ２７に供給して処理した後、再びマトリクス部５１に戻して処理を継続することができる。

【００３６】

これらの再構成可能な集積回路区画５１のさらに下方には、ストア用のアドレスを発生するのに適したデータパス部５６ｇおよび５６ｈをそれぞれ備えたエレメント５５ｇおよび５５ｈが配置されている。これらは、出力バッファ５３を介して外部デバイスにデータを出力するための制御を行う。そして、最下段には、ストア用にデータを出力するのに適したデータパス部（ＳＴ）５６ｓを備えたエレメント５５が配列されている。したがって、マトリクス部５１を用いて、エレメント５５の接続を動的に変更することにより、様々なデータフローをフレキシブルに構成でき、様々な処理を行うことができる。

【００３７】

図４（ａ）および図４（ｂ）に、ＤＩＤＬの簡単な例を示してある。図４（ａ）に示したＤＩＤＬ１０ａは２行の並列記述１１ａおよび１１ｂを有し、変数ａに変数ｂを代入する処理１２ａと、変数ｃに変数ａを代入する処理１２ｂとが、並列に動作する要素により同期して独立に行われるハードウェア構成を示している。このＤＩＤＬ１０ａにより定義されたハードウェアにおいては、あるサイクルｔ０で変数（ａ、ｂ、ｃ）が（１、２、３）であれば、次のサイクルｔ１で変数（ａ、ｂ、ｃ）は（２、２、１）となる。一方、このＤＩＤＬ１０ａの記載がプログラム１９ａだと理解すると、変数ｃは２になるので、得られる結果は異なる。しかしながら、サイクルｔ１に続く次のサイクルｔ２において、ＤＩＤＬ１０ａにより定義されたハードウェアにおいては、変数（ａ、ｂ、ｃ）は（２、２、２）となり、ＤＩＤＬ１０ａの記載がプログラム１９ａであると理解した場合と同じ結果が得られる。

【００３８】

図４（ｂ）に示したＤＩＤＬ１０ａ´は、並列記述１１ａおよび１１ｂの順番が入れ替わっているが、それぞれの記述に対応する処理１２ａおよび１２ｂが独立して行われるので各サイクルｔ０～ｔ２における演算結果は変わりない。これに対し、プログラム１９ｂであると理解すると、処理１２ａと１２ｂの順番が入れ替わるので、変数（ａ、ｂ、ｃ）は（２、２、１）となる。すなわち、処理が入れ替わったプログラムの結果は、ＤＩＤＬ１０ａまたは１０ｂのサイクルｔ１における値に合致する。したがって、ＤＩＤＬ１０ａにより、プログラム１９ａ

および１９ｂのいずれのアルゴリズムも実現するハードウェアを記述できるが、プログラム１９ａおよび１９ｂと同一の結果が得られるサイクルが異なる。そこで、本発明においては、少なくとも１つの他の並列処理の出力データを少なくとも１つの入力データとする並列処理を示す並列記述によりアルゴリズムを並列システムに割り付けると共に、複数の入力データを備えた並列記述においては、それら複数の入力データは、システムに入力されてからのレイテンシーが同一であると規定することにより、プログラムと同一のアルゴリズムにより演算が可能なハードウェア構成を定義できるようにしている。

【００３９】

図５に、ＤＩＤＬ用のコンパイラ９の概略処理を示してある。まず、ステップ３１でＤＩＤＬ１を読み込み、ステップ３２でＤＩＤＬ１に含まれた並列記述を解釈する。そして、ハードウェアライブラリ３に格納されたハードウェア構成に基づき、並列記述に示された並列処理を行う回路構成を生成する。図３に示した再構成可能なプロセッサのマトリクス部５１により並列処理を行う場合は、ハードウェアライブラリ３には、各エレメント５５の情報と、配線５７および５８とスイッチ５９の情報とが格納されており、それらのハードウェア情報にしたがって回路構成する。各エレメント５５の情報としては、例えば、演算機能、入力条件、処理サイクルがある。マトリクス部５１では、各エレメント５５の個数および配置は決まっているので、並列処理を行う回路構成としては、並列処理のために選択されたエレメントと、その位置および選択されたエレメントを接続する配線ルートといった回路情報が生成される。

【００４０】

次に、ステップ３３において、ステップ３２で生成された回路構成に複数のデータが入力される場合は、それら複数の入力データの、並列処理システム、すなわちマトリクス部５１に入力されてからのレイテンシーが一致するように遅延要素となるデータパス部（ＤＥＬ）５６ｄを備えたエレメント５５ｄを加える。さらに、ステップ３２および３３では、幾つかの最適化が行われる。ＤＩＤＬ１に含まれる並列記述により規定された並列処理が、他の並列記述により規定された並列処理の少なくとも一部と同じ共通処理を含む場合は、ステップ３２では、共

通処理に同一のエレメント５５を含む共通構成を割り当て、ステップ３３では、共通構成に含まれないエレメント５５に対して必要であれば遅延エレメント５５ｄを加えてレイテンシーを調整する。

【００４１】

そして、ステップ３４において、ステップ３３で生成された回路構成とそれらを接続する情報を含むハードウェア構成情報を出力する。これらのステップは、ＤＩＤＬ１に含まれる並列記述を個別に読み込んで行うことも可能であり、ＤＩＤＬ１に含まれる全てあるいは一部の並列記述を読み込んで行うことも可能である。

【００４２】

これらのステップ３１～３４を含む並列処理システムの生成方法は、ステップ３１～３４の各工程を汎用コンピュータ９で実行可能なコンパイラプログラムあるいはプログラム製品５として適当な記録媒体に記録して提供することができる。また、ネットワークを介してプログラム製品を提供することも可能である。そして、汎用コンピュータ９にプログラム５をインストールすることにより汎用コンピュータをコンパイラ２として使用できる。したがって、コンパイラプログラム５をインストールされたコンピュータ９は、図６に示すように、ＤＩＤＬ１を読み込む機能３５と、ハードウェアライブラリ３に基づき、ＤＩＤＬ１に記述された並列処理を実行するための回路構成を生成する機能（第１の機能）３６と、回路構成に複数のデータが入力される場合は、それら複数の入力データの、マトリクス部５１に入力されてからのレイテンシーが一致するように遅延エレメント５５ｄを加える機能（第２の機能）３７と、生成された回路構成を、エレメント間の接続情報を含めてハードウェア構成情報４として出力する機能３８を備えたコンパイラ２として動作する。

【００４３】

図７（ａ）に、異なるＤＩＤＬの例を示してある。このＤＩＤＬ１０ｂには、システム、本例ではマトリクス部５１に入力される変数を示す記述１１ｃと、内部変数を示す記述１１ｄと、足し算を示す記述１１ｅとが含まれている。このＤＩＤＬ１０ｂがコンパイラ２に読み込まれて処理されると、図７（ｂ）に示すよ

うに、算術演算が可能なデータパス５６ｃを備えた演算エレメント５５ｃを有し、その演算エレメント５５ｃに変数ｂとｃが入力され、変数ａが出力される回路構成１８ｂが生成される。

【００４４】

図８（ａ）に、さらに異なるＤＩＤＬの例を示してある。このＤＩＤＬ１０ｃには、変数に関する記述１１ｃおよび１１ｄに加え、２つの並列処理１２ｆおよび１２ｇを示す並列記述１１ｆおよび１１ｇが含まれている。このＤＩＤＬ１０ｃがコンパイラ２に読み込まれると、まずは、ステップ３３の回路構成を生成する機能において演算エレメント５５ｃを用いた回路構成が生成される。しかしながら、並列処理１２ｆおよび１２ｇは共通した処理を含んでいるので、その部分は共通構成１７ｃとされ、図８（ｂ）に示すように、並列処理１２ｇは並列処理１２ｇ´に最適化される。その結果、並列処理１２ｇ´においては、演算エレメント５５ｃに入力される変数は、システム内部で演算処理され、システムに入力される変数ｂおよびｃに対して演算エレメント５５ｃの処理サイクル分だけ後れる変数ａと、システムに入力される変数ｄとなる。したがって、ステップ３４のレイテンシーを調整する機能において、入力される変数ａと変数ｄとのレイテンシーを調整するために、システム入力される変数ｄを演算エレメント５５ｃのサイクル数分だけ遅らせる遅延エレメント５５ｄが挿入される。演算エレメント５５ｃで足し算を行った場合に消費されるサイクル数は、ハードウェアライブラリ３に格納されているので、そのサイクル数を消費するように遅延エレメント５５ｄは設定される。その結果、図８（ｃ）に示す回路構成１８ｃが生成され、複数のエレメント５５を接続するハードウェア構成情報としてコンパイラ２から出力される。なお、以降では、説明を簡単にするために、特に記載しないかぎり、エレメントにおいては１サイクルが処理されるものとして説明する。

【００４５】

図９（ａ）に、さらに異なるＤＩＤＬの例を示してある。このＤＩＤＬ１０ｄには、変数に関する記述１１ｃおよび１１ｄに加え、４つの並列処理１２ｈ～１２ｋを示す並列記述１１ｈ～１１ｋが含まれている。このＤＩＤＬ１０ｃがコンパイラ２に読み込まれると、まずは、ステップ３３の回路構成を生成する機能に

おいて演算エレメント５５ｃを用いた回路構成が生成される。並列処理１２ｋにおいては、入力される変数ｃは、システム入力変数ａを入力とする並列処理１２の出力である。また、入力される変数ｅは、システム入力変数ａおよびｂを入力とする並列処理１２ｉの出力をさらに入力とする並列処理１２ｊの出力である。したがって、ステップ３４のレイテンシーを調整する機能において、入力される変数ｃと変数ｅとのレイテンシーを調整するために、変数ｃを遅らせる遅延エレメント５５ｄが挿入される。その結果、図９（ｂ）に示す回路構成１８ｄが生成され、複数のエレメント５５を接続するハードウェア構成情報としてコンパイラ２から出力される。

【００４６】

図９（ａ）の記述をプログラム１９ｄとして考えた場合、処理１２ｈ～１２ｋは時間軸に従って上から順番に行われる。したがって、処理１２ｋで演算される変数ｃおよび変数ｅは、それぞれ、先行する処理１２ｈおよび１２ｊによりそれぞれ決定されたものとなる。しかしながら、図９（ａ）の記述を単に並列実行される複数の処理の記述であると考えると、並列処理１２ｋの変数ｃおよび変数ｅは、１サイクル前に定まった値となり、プログラム１９ｄとして考えた場合と処理１２ｋの出力が異なる。これに対し、図９（ａ）の記述が本発明の定義ファイル、すなわちＤＩＤＬ１０ｄであるとすると、並列処理１２ｋの変数ｃおよび変数ｅは、レイテンシーが同一であると解釈される。したがって、２サイクル前に確定した変数ｃが遅延エレメント５５ｄを介して１サイクル前に確定した変数ｅと同期を取って並列処理１２ｋに入力される。その結果、並列処理１２ｋの出力ｆは、変数ａおよびｂを入力してから３サイクル後にプログラム１９ｄと同じになり、並列処理を記述したＤＩＤＬ１０ｄによりプログラム１９ｄと同じ結果が得られることが分かる。

【００４７】

並列処理を記述したＤＩＤＬ１０ｄは、それに含まれる記述１１ｈ～１１ｋの順番を変えても処理される内容は同じである。これに対し、プログラム１９ｄは、処理がならんだ順番、すなわち、時系列でアルゴリズムを定義しているので、記述を入れ替えると処理される内容は異なる。しかしながら、複数の並列記述に

対して入力変数のレイテンシーが同一になるという定義を導入することにより、プログラムとして記述されたファイルを、並列処理を定義するファイル、すなわち、ハードウェアを定義するファイルとして理解することが可能となる。この結果、プログラマーは、普通に時系列に従ってアルゴリズムを示したプログラムを記述することにより、並列処理システムのハードウェアを記述する本発明の定義ファイルを作成することが可能となる。したがって、本例のＤＩＤＬ１を用いることにより、プログラマーはプログラムを作成するのと同じ感覚でハードウェア設計を行うことができる。

【００４８】

さらに、定義ファイルでは、並列処理に入力される変数のレイテンシーが同一と理解されるだけであり特定のハードウェアを前提としない。すなわち、レイテンシーが同一の入力変数は、同期して、あるシステム（本例ではマトリクスユニット５１）にロードされたデータ群に含まれるものか、またはその同期してロードされたデータ群のいずれかが並列処理により処理されたデータ群に含まれものである、同期してシステムに入力された変数由来であること以外は、実際のハードウェアを前提として設計するとき以外に理解する必要がない。したがって、本発明のＤＩＤＬは、ハードウェアに依存しないハードウェア記述言語であると言うことができる。すなわち、どのようなハードウェアを前提としても解釈することができ、ハードウェアが特定されれば、そのハードウェアに定義ファイルに記述されたアルゴリズムを割り付けることができる。したがって、本例のＤＩＤＬ１は、時間的ではなく、空間的にアルゴリズムを展開できる言語ということができる。このため、プログラムで記述できるアルゴリズムは、本例のＤＩＤＬ１を用いることにより、すべて、並列に動作する複数の要素、本例であればエレメント５５を組み合わせた回路により実行させることができ、ＤＩＤＬ１により、実行するためのハードウェア構成を記述することができる。

【００４９】

図１０に、さらに異なるＤＩＤＬの例を示してある。このＤＩＤＬ１０ｅには、外部から与えられる所定の数（ｎｕｍ０ｆＤａｔａ）の入力変数ｉｎの最大値ａを検索する処理が記述されている。初期値をセットする記述１１１に続く部分

が並列処理を記述した部分であり、入力変数ｉｎとそれまでの最大値ａとを比較して最大値をセットする処理１２ｍの記述１１ｍと、カウンターを進める処理１２ｎの記述１１ｎと、処理１２ｎのカウントアップした値が所定の数ｎｕｍ０ｆＤａｔａに達したら処理１２ｍの最大値ａを出力する処理１２ｏの記述１１ｏとを備えている。したがって、処理１２ｏにおいて、システムへ入力される変数ｉｎに対してレイテンシーが調整される。

【００５０】

図１１に、このＤＩＤＬ１０ｅがコンパイルされたハードウェア構成１８ｅを示してある。ステップ３３の回路構成を生成する機能において、処理１２ｍは、演算エレメント５５ｃを用いて回路構成され、処理１２ｎは、アドレス生成用のエレメント５５ｂを用いて回路構成され、処理１２ｏは２つの演算エレメント５５ｃを用いて回路構成される。処理１２ｍを行う演算エレメント５５ｃにおいて消費されるサイクル数が、処理１２ｎのカウント処理を行うエレメント５５ｂにおいて消費されるサイクル数より多いので、遅延エレメント５５ｄが処理１２ｏのカウンター値を入力する側に追加される。これにより、処理１２ｏに入力される、処理１２ｍの出力と処理１２ｎの出力のレイテンシーが調整される。なお、処理１２ｏの出力側に配置された２つの遅延エレメント５５ｄは、マトリクス部５１が３つのセグメントに分かれており、最初のセグメントで処理１２ｍ～１２ｏの回路が構成されるので、他のセグメントを通過して出力用のエレメント５６ｓにデータを転送するためのものである。

【００５１】

図１２に、図３に示したマトリクスユニット５１に配置されたエレメント５５に図１１に示した回路構成を割り付けた状態を示してある。本例においては、エレメント５５が既にマトリクス状に配置された再構成可能なプロセッサ２０にＤＩＤＬ１０ｅのアルゴリズムを割り当てている。したがって、図１１に示したハードウェア構成の情報は、図１２に示したように、選択されたエレメントの配置と、それらを接続する配線ルートの情報としてコンパイラ２で生成され、ＤＤＤＬ４として出力される。なお、アドレスを生成するエレメント５５ａおよび５５ｂは、入力変数ｉｎを外部から入力するために使用されている。また、本図には

示していないが、出力用のアドレスを生成するエレメント５５ｇおよび５５ｈも同様に使用されている。

【００５２】

本例の並列処理システム２０は、エレメント５５がマトリクス状に配置された再構成可能なプロセッサ（集積回路装置）であり、各エレメント５５は、上述したように、ＡＬＵなどのある程度の規模の演算機能を備え、データを８ビット、１６ビットあるいは３２ビットなどのバイトあるいはワード単位で、特定の目的のために処理するのに適した演算ユニットである。そして、入力データと出力データとをフリップフロップなどを使ってラッチし、クロック信号で同期している。すなわち、各エレメント５５における入力と出力はクロックで同期されている。したがって、各エレメント５５で消費されるサイクルは予め予想することができる。さらに、各エレメントは特定の処理を行うのに適したある程度の規模の演算機能を備えているので、ＤＩＤＬ１に記述された並列処理にエレメントの単位でハードウェアを割り当てて回路構成することができる。したがって、本例の並列処理システム２０を前提としてＤＩＤＬ１を解釈すると並列処理で消費されるサイクルは容易に予想でき、回路構成を生成するのが容易であり、さらにレイテンシー調整も容易に行える。このため、ある程度の規模の演算能力を備えて、１つの並列処理をエレメント単位でほぼ実行できるデータ処理装置２０は、本発明の定義ファイルからハードウェアを生成するのに適したアーキテクチャであると言える。

【００５３】

また、エレメント５５の単位で並列処理が割り付けられ、サイクル数も管理されるので、ＤＩＤＬ１で定義されたアプリケーションを実行している途中で、あるエレメント５５の機能を外部入力により変えても、他の並列処理に予期しない影響を及ぼさずにアプリケーションを実行させることができる。機能を変えたエレメント５５におけるサイクル数が変動するのであれば、それに対応できるように遅延エレメント５５ｄを接続しておき、遅延エレメント５５ｄの内部のサイクル数を同様に外部入力により変更し、他のエレメント５５で形成されたデータフローに影響を及ぼさずに特定のエレメント５５の処理をダイナミックに変更する

ことが可能である。

【００５４】

図１３に、ＤＩＤＬレベルのシミュレータ６７を示してある。このシミュレータ６７は汎用のコンピュータ９にシミュレータ用のプログラム６８をインストールすることにより構成される。したがって、本発明の定義ファイルであるＤＩＤＬ１に基づき、並列に動作する複数種類の要素を備えたシステムをコンピュータ９によりシミュレートするためのプログラムあるいはプログラム製品６８も、ＣＤ－ＲＯＭなどの適当な記録媒体に記録したり、コンピュータネットワークを介して提供することができる。

【００５５】

図１４にシミュレータ６７の概略動作をフローチャートにより示してある。まず、ステップ７１でＤＩＤＬ１を読み込む。次に、ステップ７２で、ＤＩＤＬ１に記述された複数の並列処理を同期して実行する。この際、複数の入力データを備えた並列記述により指示された並列処理を実行する場合は、それら複数の入力データに、システムに入力されるデータとして定義されているデータに対するレイテンシーが同一のデータを用いる。ステップ７３で終了条件、例えば、所定の回数、並列処理を繰り返して実行したり、ＤＩＤＬ１に記述された並列処理の結果が所定の値に達するなどの条件が成立すると、ステップ７４でシミュレーションした結果を出力する。これにより、ハードウェアに依存しないで、ＤＩＤＬ１に記述されたハードウェアの動作をシミュレートすることができる。

【００５６】

【発明の効果】

以上に説明したように、本発明においては、同期して独立に行われる複数の並列処理をそれぞれ規定した複数の並列記述を有する定義ファイルであって、複数の入力データを備えた並列記述においては、それら複数の入力データは、システムに入力されてからのレイテンシーが同一であると解釈する定義ファイルを提案している。この定義ファイルは、並列処理を記述するものであり、ハードウェア記述ファイルであると理解することができ、また、そこにはハードウェア自体は表れないのでハードウェア依存性のないハードウェア記述であると言うことがで

きる。さらに、並列記述であるので、プログラムカウンタの必要のない形態でアルゴリズムを定義することができる。

【００５７】

したがって、本発明の定義ファイルにより、従来の高級言語と類似した記述ではあるが、時間順序性のない並列記述により、アルゴリズムに含まれる順序を時間的にではなく空間的に記述でき、並列に動作する複数の要素を備えた並列処理システムの生成を短期間で容易に行うことが可能となる。特に、ＡＬＵなどのある程度の規模の演算機能を備えた演算ユニットがマトリクス状に配置された並列処理システム、さらには、演算ユニットの接続を変えられる再構成可能なデータ処理システムを設計したり開発したりするのに本発明の定義ファイルは有用である。

【図面の簡単な説明】

【図１】

定義ファイルであるＤＩＤＬからハードウェア構成情報であるＤＤＤＬを生成する概略構成を示す図である。

【図２】

再構成可能な並列処理システムの概要を示す図である。

【図３】

独立に並列に動作する複数のエレメントがマトリクス状に配置された並列処理システムを示す図である。

【図４】

ＤＩＤＬの例である。

【図５】

コンパイラの概略処理を示すフローチャートである。

【図６】

コンパイラの概略構成を示すブロック図である。

【図７】

ＤＩＤＬの例と、それに対応する回路構成を示す図である。

【図８】

ＤＩＤＬの他の例と、それに対応する回路構成を示す図である。

【図９】

ＤＩＤＬのさらに異なる例と、それに対応する回路構成を示す図である。

【図１０】

ＤＩＤＬのさらに異なる例を示す図である。

【図１１】

図１０に示すＤＩＤＬに対応する回路構成を示す図である。

【図１２】

図１１に示す回路構成をマトリクスユニットに割り付けた状態を示す図である。

【図１３】

ＤＩＤＬレベルでシミュレートする概略構成を示す図である。

【図１４】

ＤＩＤＬレベルのシミュレータの処理の概要を示すフローチャートである。

【符号の説明】

1、１０ａ～１０ｅ　ＤＩＤＬ（定義ファイル）

2　　コンパイラ

3　　ハードウェアライブラリ

4　　ＤＤＤＬ（ハードウェア構成情報）

１１ａ～１１ｏ　　並列記述

１２ａ～１２ｏ　　並列処理

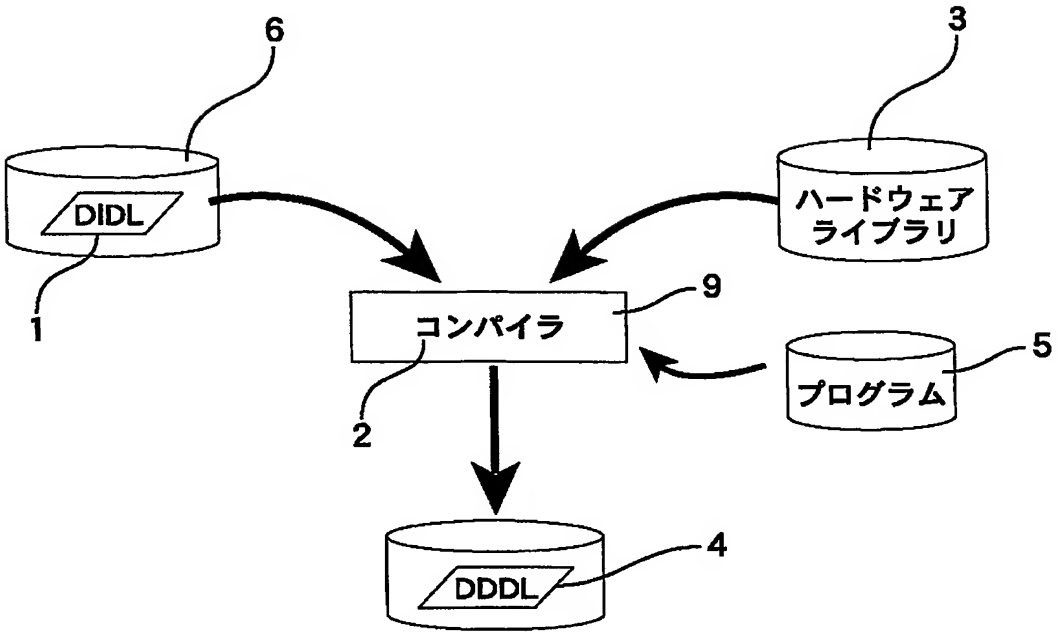
２０　　並列処理システム

５１　　マトリクスユニット

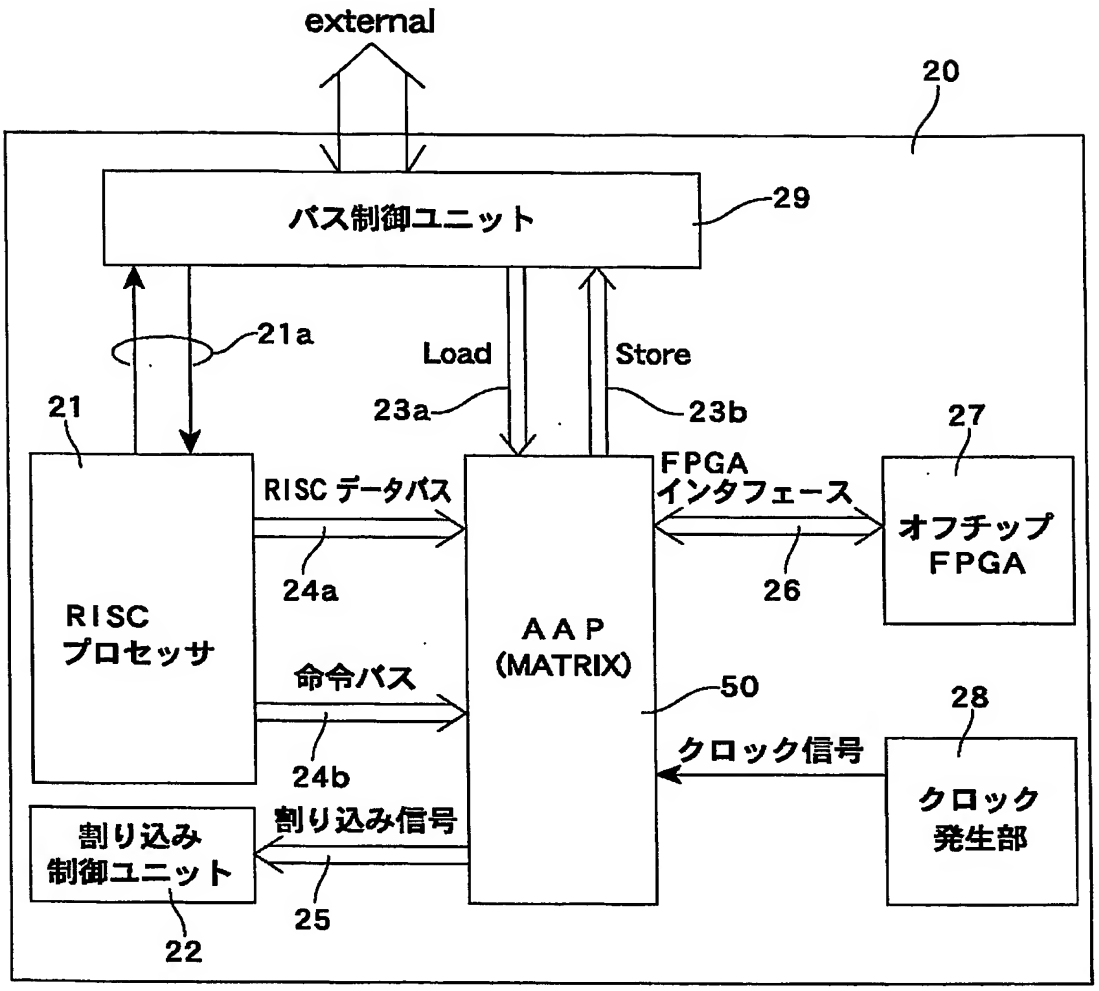
５５　　エレメント

【書類名】　図面

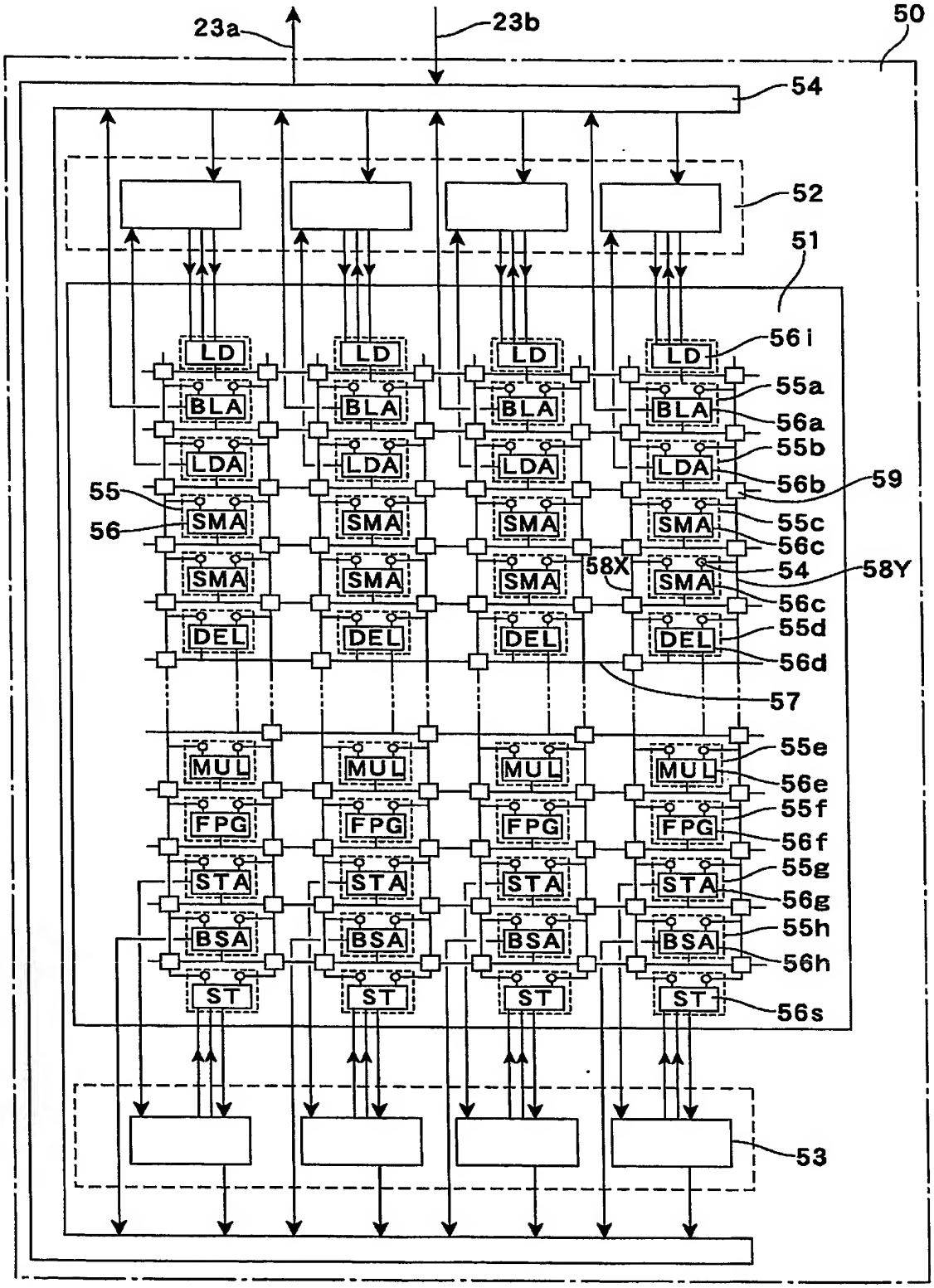
【図１】



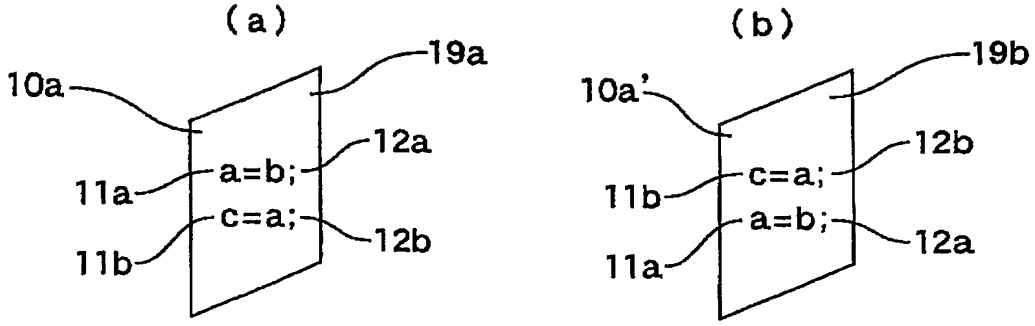
【図２】


external
20
バス制御ユニット
29
21a
Load
Store
23a
23b
21
27
RISC データバス
FPGA
インタフェース
オフチップ
FPGA
24a
26
RISC
プロセッサ
AAP
(MATRIX)
命令バス
50
28
クロック信号
24b
クロック
発生部
割り込み
制御ユニット
割り込み信号
25
22

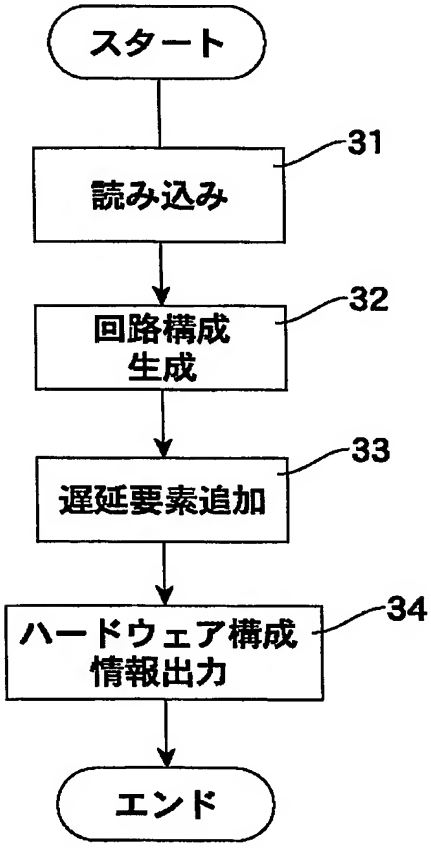
【図３】


23a
23b
50
54
52
51
LD
BLA
LDA
SMA
DEL
MUL
FPG
STA
BSA
ST
56i
55a
56a
55b
56b
59
55
56
55c
56c
58X
54
58Y
56c
55d
56d
57
55e
56e
55f
56f
55g
56g
55h
56h
56s
53

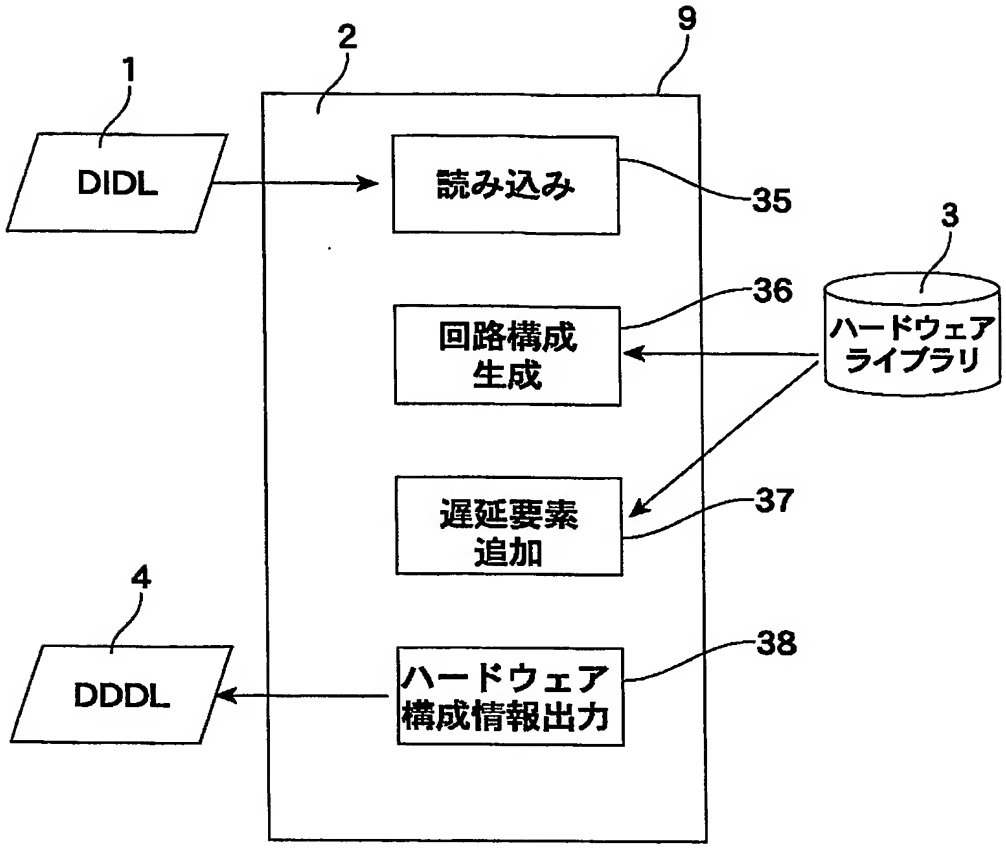
【図４】


（a）
10a
19a
12a
a=b;
11a
c=a;
12b
11b
（b）
10a'
19b
12b
c=a;
11b
a=b;
12a
11a

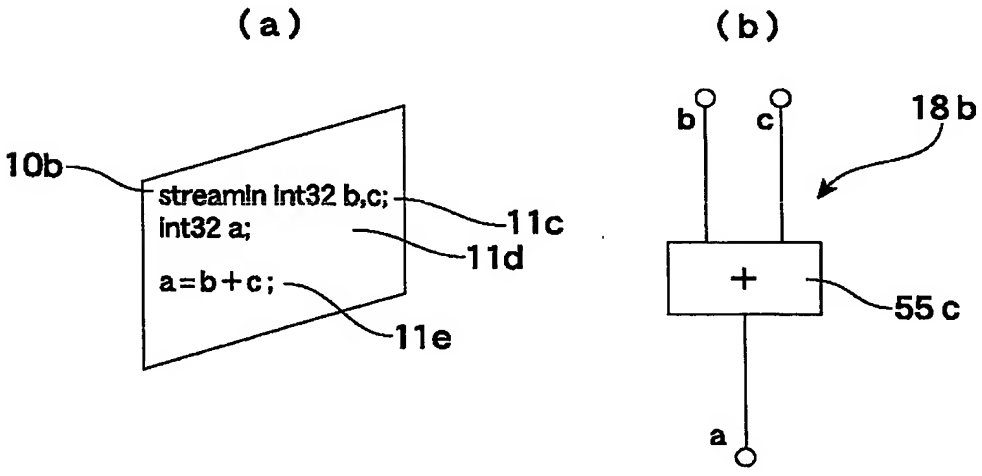
【図５】


スタート
31
読み込み
32
回路構成
生成
33
遅延要素追加
34
ハードウェア構成
情報出力
エンド

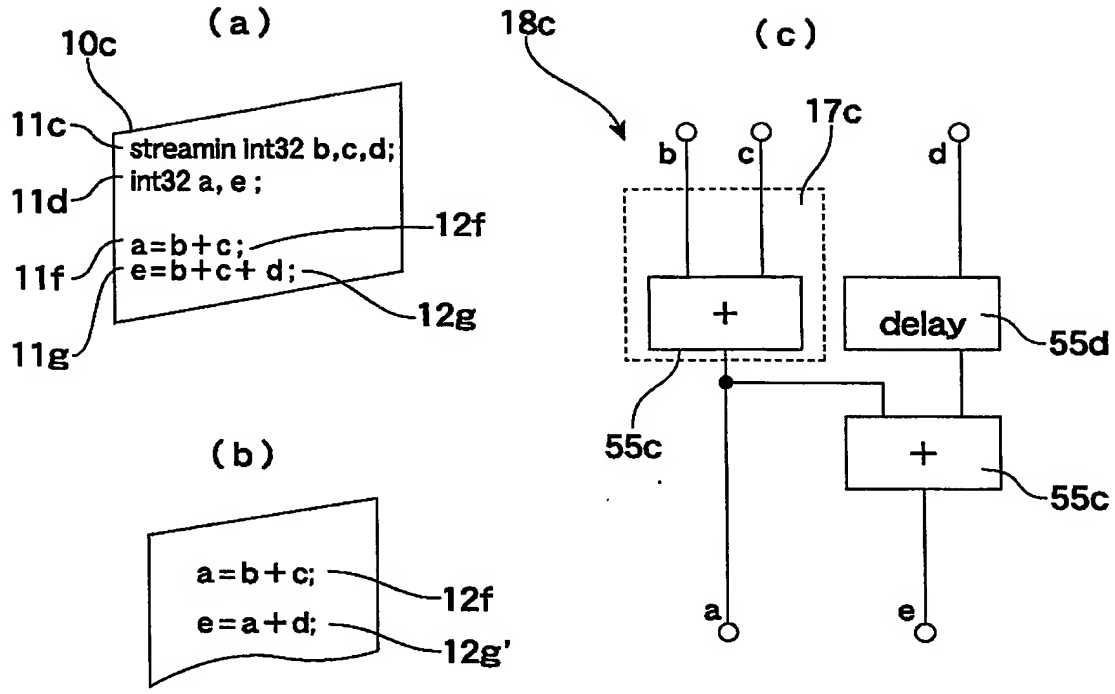
【図６】


1
2
9
DIDL
読み込み
35
3
36
回路構成
生成
ハードウェア
ライブラリ
遅延要素
追加
37
4
DDDL
ハードウェア
構成情報出力
38

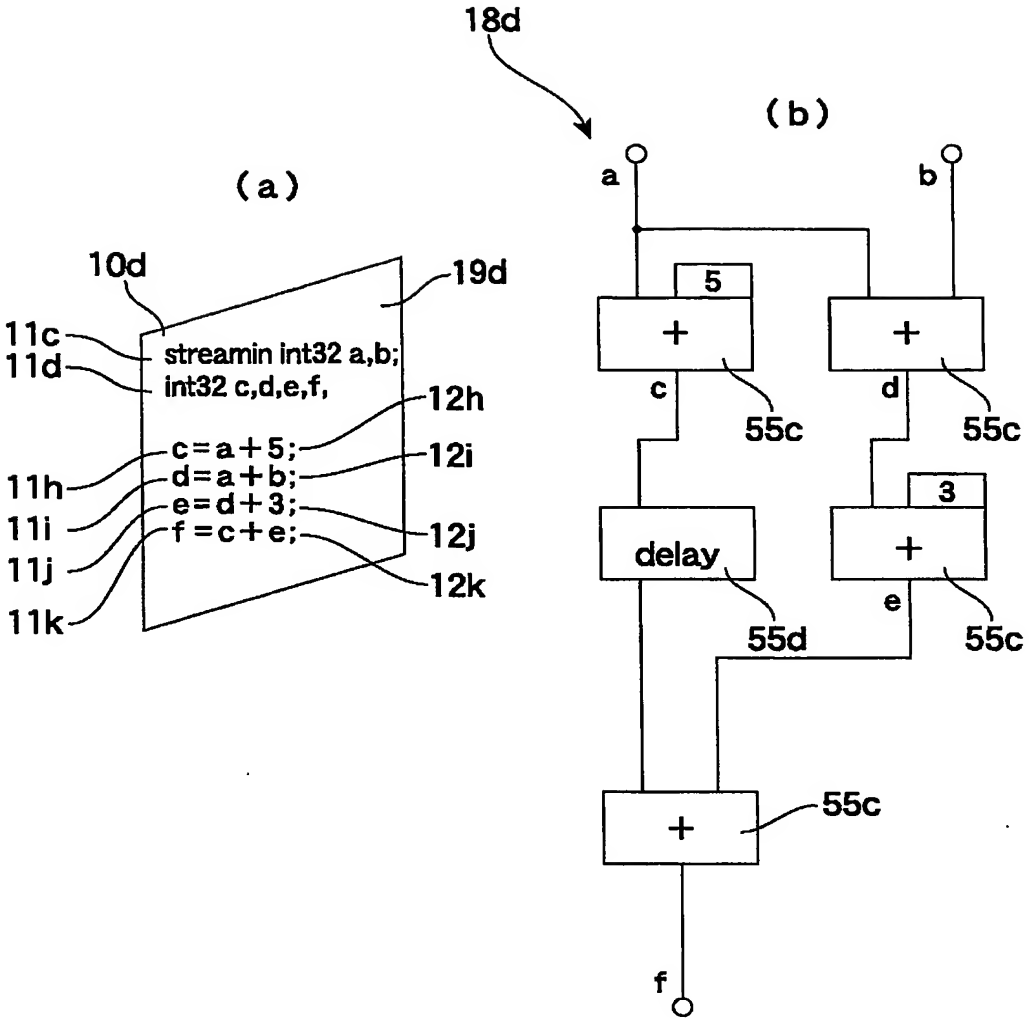
【図７】


(a)
(b)
10b
streamIn int32 b,c;
int32 a;
a=b+c;
11c
11d
11e
b
c
18b
+
55c
a

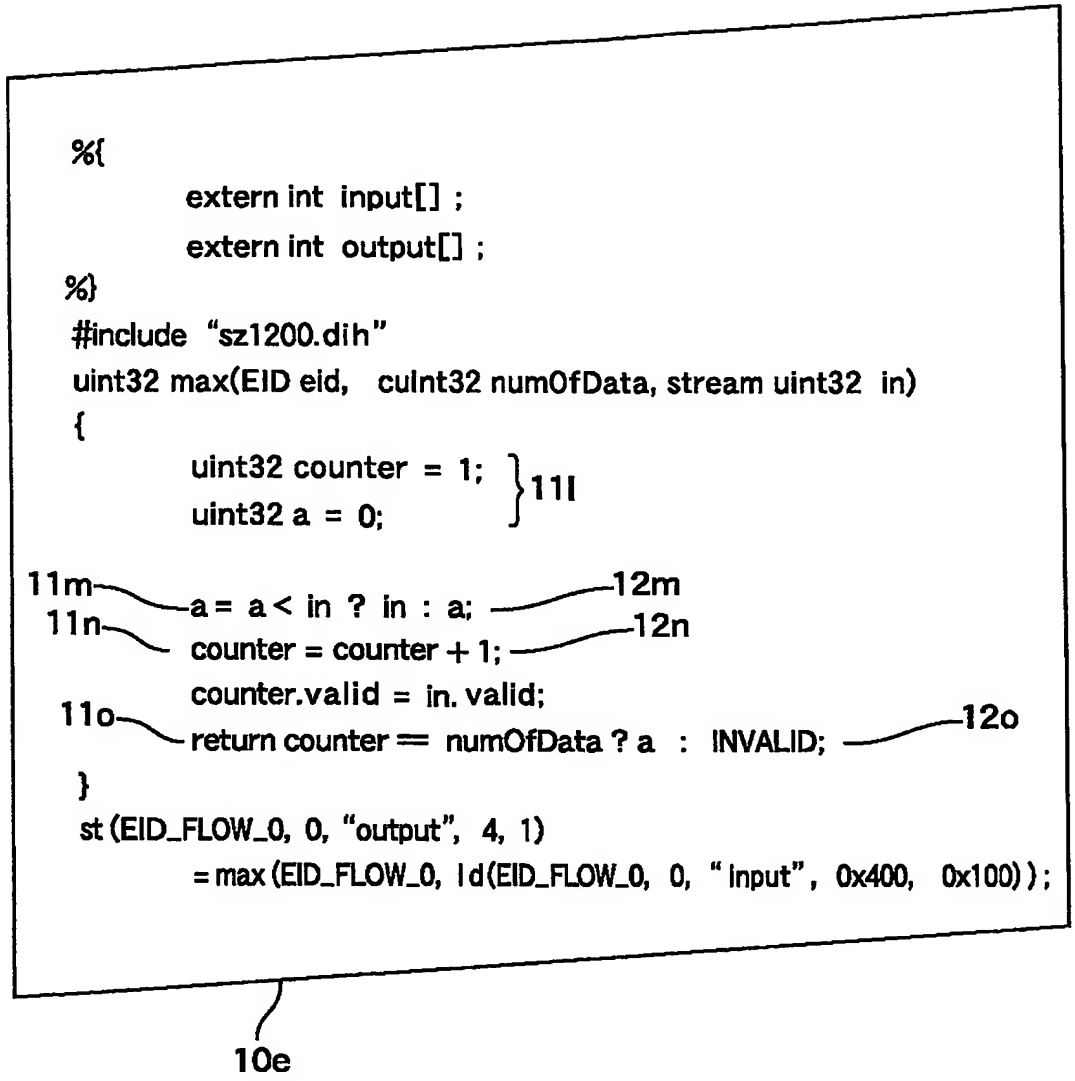
【図８】


(a)
10c
11c
streamin int32 b,c,d;
int32 a, e ;
11d
a=b+c;
12f
11f
e=b+c+ d;
12g
11g
(b)
a=b+c;
12f
e=a+d;
12g'
18c
(c)
b
c
17c
d
+
delay
55d
55c
+
55c
a
e

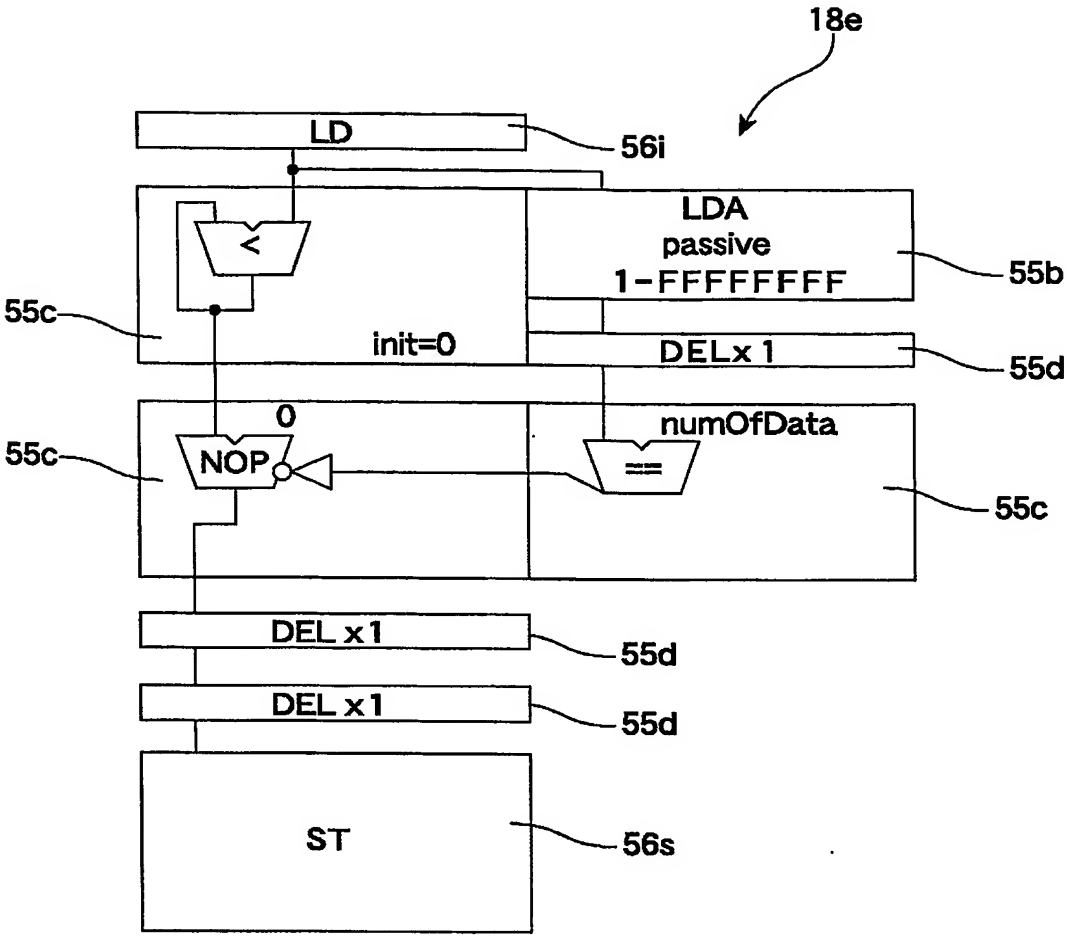
【図９】


18d
(a)
(b)
10d
19d
11c
11d
streamin int32 a,b;
int32 c,d,e,f,
12h
c=a+5;
12i
d=a+b;
11h
11i
e=d+3;
12j
11j
f=c+e;
12k
11k
a
b
5
+
+
c
55c
d
55c
3
delay
+
e
55d
55c
+
55c
f

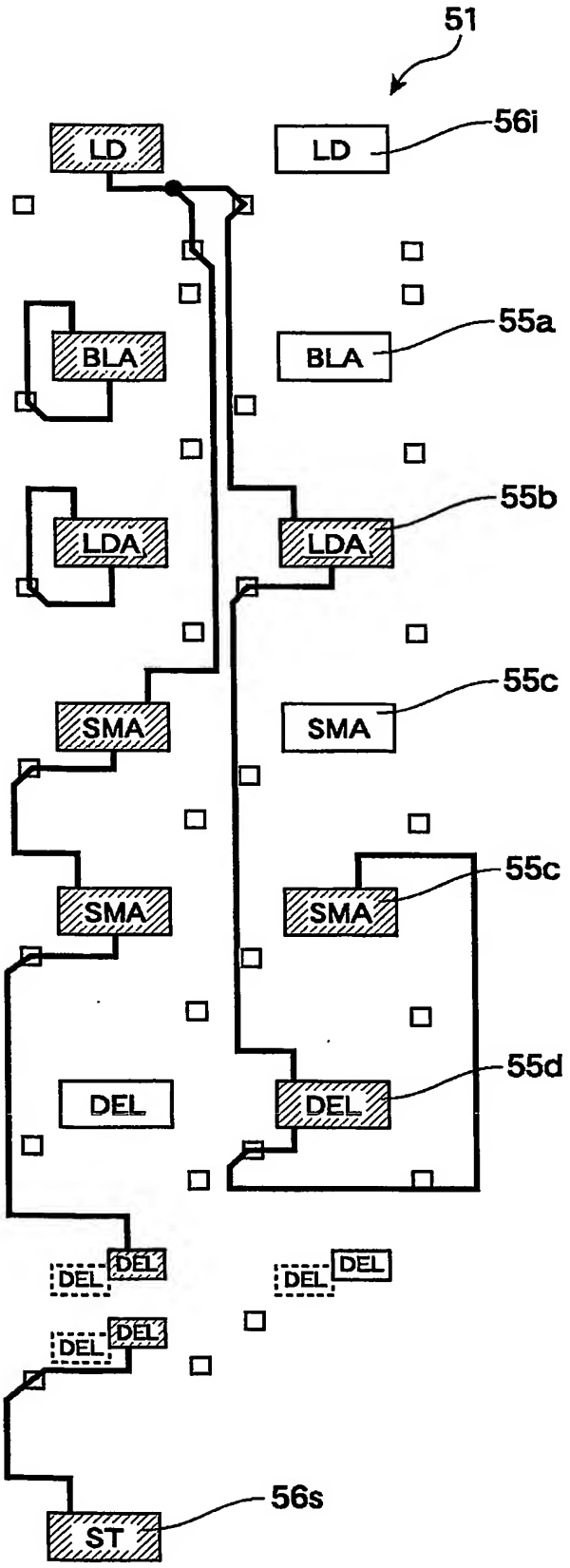
【図１０】


%{
extern int input[] ;
extern int output[] ;
%}
#include "sz1200.dih"
uint32 max(EID eid, cuInt32 numOfData, stream uint32 in)
{
uint32 counter = 1;
uint32 a = 0;
11l
11m
a = a < in ? in : a;
12m
11n
counter = counter + 1;
12n
counter.valid = in.valid;
11o
return counter == numOfData ? a : INVALID;
12o
}
st (EID_FLOW_0, 0, "output", 4, 1)
= max (EID_FLOW_0, Id(EID_FLOW_0, 0, "Input", 0x400, 0x100));
10e

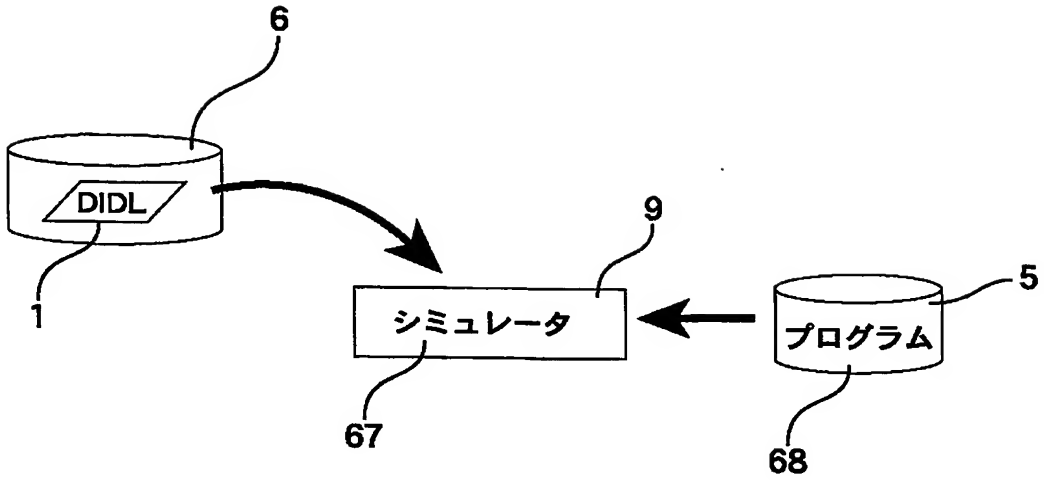
【図１１】


18e
LD
56i
LDA
passive
1-FFFFFFFF
55b
<
55c
init=0
DELx 1
55d
0
NOP
numOfData
==
55c
55c
DEL x1
55d
DEL x1
55d
ST
56s

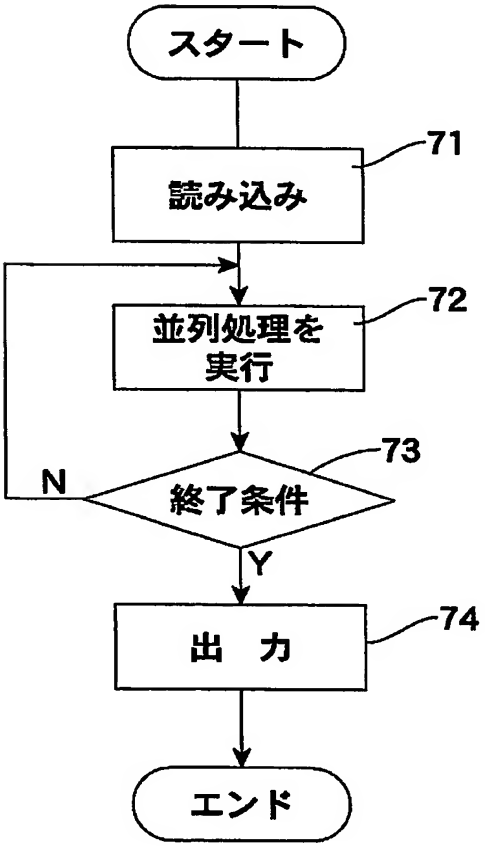
【図１２】


51
LD
LD
56i
BLA
BLA
55a
LDA
LDA
55b
SMA
SMA
55c
SMA
SMA
55c
DEL
DEL
55d
DEL
DEL
DEL
DEL
DEL
ST
56s

【図１３】


6
DIDL
1
9
シミュレータ
67
5
プログラム
68

【図１４】


スタート
71
読み込み
72
並列処理を
実行
73
N
終了条件
Y
74
出　力
エンド

【書類名】　要約書

【要約】

【課題】　並列に独立して動作する複数のエレメントを備えた並列処理システムの開発に適したハードウェア記述手段を提供する。

【解決手段】　同期して独立に行われる複数の並列処理をそれぞれ規定した複数の並列記述を有する定義ファイルであって、複数の入力データを備えた並列記述においては、それら複数の入力データは、システムに入力されてからのレイテンシーが同一であると解釈する定義ファイル（ＤＩＤＬ）１を提供する。コンパイラ２においては、ＤＩＤＬ１を読み込む機能３５と、ハードウェアライブラリ３を参照して回路構成を生成する機能３６と、入力データのレイテンシーを調整する機能３７とを備えており、ＤＩＤＬ１により定義されたハードウェア構成情報を出力することができる。

【選択図】　図６

認定・付加情報

特許出願の番号	特願２００３－１８５４８１
受付番号	５０３０１０８０１３２
書類名	特許願
担当官	第七担当上席　　００９６
作成日	平成１５年　６月３０日

＜認定情報・付加情報＞

【提出日】　平成15年 6月27日

特願２００３－１８５４８１

出　願　人　履　歴　情　報

識別番号　　[５００２３８７８９]

１．変更年月日　　２００３年　６月　２日
[変更理由]　　住所変更
住　所　　東京都品川区上大崎二丁目２７番１号
氏　名　　アイピーフレックス株式会社